Fonctionnement asynchrone et Ajax

OBJECTIF

Ce TD a pour objectif de vous faire manipuler du code JavaScript asynchrone.

Rappel : Si des informations ne sont pas présentes dans ce sujet, c'est certainement qu'elles ont été données lors des TD précédents. Reportez-vous aux sujets des TD 1, 2, 3 et 4 ainsi qu'à vos notes. Vous ne pouvez pas dire que vous n'étiez pas au courant qu'il fallait prendre des notes, je n'ai de cesse de vous l'écrire.

QUESTION 1

Commencez par recenser le nombre d'étudiants de votre groupe qui ont pris de notes au cours des précédents TD et TP.

QUESTION 2

Pour démarrer avec le fonctionnement asynchrone de JavaScript, vous allez étudier le chargement et l'affichage d'une image. Cela se passera en quatre temps :

- Sélection du fichier image,
- Lecture du contenu du fichier,
- Création et initialisation de l'objet Image
- Affichage de l'image

Etape 1 – Sélection du fichier

Pour la première étape, rien à faire, le navigateur s'occupe de tout.

Etape 2 – Lecture du fichier

Concernant la lecture du fichier, vous l'avez déjà fait lors du TD 3, mais je suis certain qu'au vu du résultat de la question 1, un petit rappel ne sera pas superflu.

- Créez un fichier 'question2.js' dans lequel vous déclarerez la fonction 'lireFichier' qui lira le contenu du fichier sélectionné grâce à l'objet FileReader :

```
var fileReader = new FileReader();
fileReader.readAsDataURL(fichier); //Lecture du fichier
var contenuFichier = fileReader.result; //Contenu du fichier lu
```

- Affichez dans la console le contenu du fichier lu.
- Connectez la fonction 'lireFichier' au bouton 'btnChargerImage'.
- Testez. Qu'affiche la console ?

Ce qui se passe est assez simple, vous essayez d'afficher le contenu lu dans le fichier alors que la lecture n'est pas terminée, voire qu'elle n'a pas démarré. Il va donc falloir attendre que la lecture du fichier soit terminée avant de chercher à en afficher le contenu.

L'objet FileReader dispose d'une propriété 'onload' à laquelle vous pouvez associer une fonction d'appel (callback). Cette fonction sera appelée sitôt que la lecture du fichier sera terminée.

```
var fileReader = new FileReader();

fileReader.onload = function() {
     var contenuFichier = fileReader.result; //Contenu du fichier lu
};

fileReader.readAsDataURL(fichier); //Lecture du fichier
```

- Modifiez votre code afin d'intégrer une fonction d'appel qui affichera le contenu du fichier dans la console.

  Note : il est vivement conseillé d'affecter la propriété 'onload' avant d'entamer la lecture du fichier. Sans quoi, si le fichier est petit, la lecture pourrait être terminée avant que la fonction d'appel ne soit associée. Celle-ci ne serait donc jamais appelée.

- Testez.

Etape 3 – Chargement de l'image

Maintenant que nous disposons du contenu du fichier image, nous allons pouvoir le transmettre à un objet Image afin de l'intégrer dans notre page Web.

Déclarez une fonction 'chargerImage' qui prendra en paramètre les informations de l'image à charger au format DataURL (comprenez le contenu du fichier précédemment lu). Un objet Image sera créé et initialisé à partir de ces informations :

```
var image = new Image();
image.src = dataURL;
```

Etape 4 – Affichage de l'image

Dernière étape, afficher l'image !

- Créez une fonction 'afficherImage' qui prendra comme paramètre l'image que l'on souhaite affichée. Utilisez JQuery pour affecter les données de l'image comme arrière-plan de l'élément 'aperçuImage' de la page Web :

  ```
  //Code CSS
  background-image: url( dataURL );
  ```

- La fonction 'afficherImage' affichera également les dimensions de l'image dans le paragraphe 'dimensions-image'.
- Chainez l'appel de vos fonctions : 'lireFichier' devra appeler 'chargerImage' qui appellera 'afficherImage'.
- Testez. Que se passe-t-il ? (Utilisez Firefox. Chrome et Edge sont des chenapans qui ne permettent pas de voir ce que je veux vous montrer…).

Comme la lecture d'un fichier, le chargement d'une image prend du temps. L'image s'affiche correctement car les données sont déjà prêtes. Mais les dimensions de l'image sont calculées une fois que l'image est chargée.

- Modifiez le code de la fonction 'chargerImage' et utilisez la propriété 'onload' de l'image pour appeler la fonction 'afficherImage'.
- Testez.

QUESTION 3

A présent, voyons un autre cas de fonctionnement asynchrone. Vous l'avez vu en cours, JavaScript peut effectuer des requêtes auprès d'un serveur afin d'obtenir des informations ou d'en stocker côté hébergement. Cependant, lorsque vous effectuez de telles requêtes, vous ne savez pas au bout de combien de temps le serveur va vous répondre. Voyons comment cela fonctionne !

Important : cette question nécessite l'utilisation d'un serveur local (wamp, xamp) ou distant (serveur de PHP).

- Créez un fichier 'question3.js' dans lequel vous déclarerez une fonction 'envoyerNote' qui :
    - récupérera les informations saisies (prénom et note)
    - transmettra ses informations au serveur via une requête au fichier 'envoyer-note.php' (dossier php/scripts). Le données devront être présentées au format JSON, dans un tableau associatif de la forme suivante :

```
var info = {
     nom: 'nom',
     note: 10
};
```

    - affichera la réponse du serveur dans le paragraphe 'reponseServeur'.

L'objet de base de JavaScript pour effectuer une requête auprès du serveur est XMLHttpRequest :

```
var xhr = new XMLHttpRequest();
xhr.onreadystatechange = function()
{
      //Si on reçoit une réponse du serveur
      if (xhr.readyState == 4 && (xhr.status == 200 || xhr.status == 0))
      {
            //Reponse du serveur
            var reponse = xhr.responseText;
      }
};

//Script à exécuter
xhr.open("POST", 'url.php', true);

//Informe le serveur du type d'information envoyée
xhr.setRequestHeader("content-type", "application/x-www-form-urlencoded");

//Exécution de la requête avec passage de données éventuel
xhr.send('data=' + donneesAEnvoyer);
```

- Connectez la fonction 'envoyerNote' au bouton 'btnEnvoyerNote'.
- Testez.

QUESTION 4

Même exercice que précédemment, mais avec une petite touche de simplification. Ajax est une surcouche à XMLHttpRequest qui a pour but de rendre plus simple son utilisation. Et JQuery à intégrer cette surcouche dans ces fonctionnalités.

- Créez un fichier 'question4.js' dans lequel vous déclarerez une fonction 'envoyerNoteAjax' qui fera la même chose que 'envoyerNote' à la différence qu'à la place de XMLHttpRequest, vous utiliserez :

```
$.ajax({
	url: 'url.php', //Script à exécuter
	method: 'POST', //Méthode d'envoi des données (POST ou GET)
	data: 'data=' + JSON.stringify(data), //Données à transmettre
	success: function (resultat) //Réponse du serveur si tout s'est bien passé
	{
		//résultat contient la réponse du serveur
	},
	error: function(error)
	{
		//Une erreur s'est produite
	}
});
```

- Testez.
- Plus simple et plus lisible, non ?

QUESTION 5

Le serveur nous dit avoir enregistré nos notes, soit, mais il serait bon de vérifier tout de même. Pour cela vous allez écrire les scripts JavaScript et PHP permettant :

- d'afficher la liste des notes saisies
- de supprimer toutes les notes

Pour cela, utilisez la classe PHP Notes située dans le dossier 'php/models', ainsi que votre tête. Et pensez à démarrer une session dès le début de vos scripts PHP.

A présent, voyons ce qu'il se passe lorsqu'on vous lâche la main...